interweb

Generated on 13/11/2026 19:30

KNOWLEDGE BASE · PUBLIC DOCUMENT

Testing SMS verification in the Sandbox: the code comes back in the response

WhatsApp & APIs

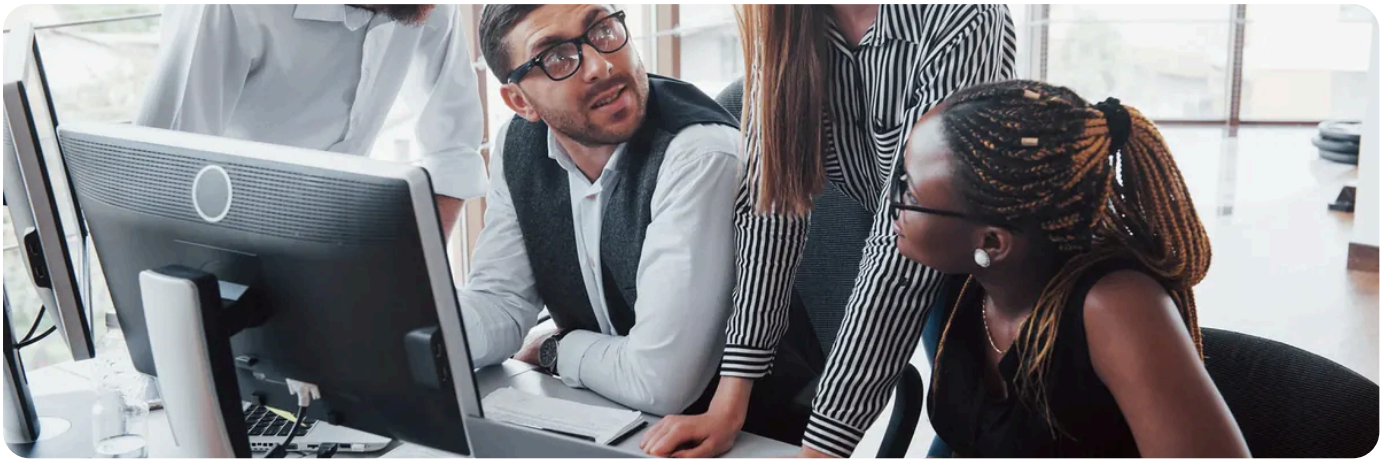
CODE	VERSION	COUNTRY · LANGUAGE
RLX-MZ-KB-2652	v2026.11.13	Mozambique · English

Interweb Mozambique
RLX Holdings, S.A. · Group RLX Holdings
Tax ID (NUIT) 401 191 607 · Legal entity no. (NUEL) 105 045 974 · Economic activity code (CAE) 62022

https://www.meu.interweb.co.mz/knowledgebase/2652

Testing SMS verification in the Sandbox: the code comes back in the response

WhatsApp & APIs Created on **November 13, 2026** 3 min read



The short answer: with a `sandbox_` key, the verification API does everything except send: it checks the data, creates the code and returns it in the `code` field of the response. No SMS goes out and no credits are used. That lets you test your system's checks, errors, expiry and resends without a phone in your hand. When everything works, swap the key for a `prod_` one: the address and the requests stay the same.

Before you start

1. **Activate SMS** on the SMS page in Meu Interweb, if you have not yet. The verification API needs the SMS service to be active.

2. **Create a Sandbox key** on the API page in Meu Interweb, with `sms:write` (send and check) and, if you want to read statuses and rules, `sms:read`. The key is shown only once.

3. **Make the first send** and find the code in the response.

```
curl https://api.interweb.co.mz/v1/sms/otp \
-H "Authorization: Bearer sandbox_..." \
-H "Content-Type: application/json" \
-H "Idempotency-Key: signup-7f3a9c" \
-d '{"number": "841234567"}'
```

The response includes `"simulated": true`, `"mode": "sandbox"` and the `code`, plus the `X-API-Simulated: true` header. Now check it:

```
curl https://api.interweb.co.mz/v1/sms/otp/otp_3f9a1c.../verify \
-H "Authorization: Bearer sandbox_..." \
-H "Content-Type: application/json" \
```

```
-d '{"code": "482913"}'
```

Tests worth running

Test	How	What you should see
Right code	Send the `code` from the response	`valid: true`, status `verified`
Wrong code	Send other digits	`valid: false` and `attempts_left` going down
Attempts used up	Keep getting it wrong, with a low `max_attempts`	Status `failed`; the right code no longer works
Expired code	Send with `expires_in` set to 1 and wait a minute	`valid: false`, status `expired`
Resend	`POST /v1/sms/otp/{id}/resend`	A new `code` in the response; the old one stops working
Replaced code	Request two codes in a row for the same number	The first becomes `cancelled`
Double click	Repeat the send with the same `Idempotency-Key`	The same response, no new code

Never pass the `code` to the browser, not even in the Sandbox.
Read it in your server logs or your test program. A code that travels to the page "just for testing" ends up forgotten in Production, and then anyone can confirm any number. In Production the `code` field simply is not there.

Moving to Production without surprises

Create a `prod_` key with the same permissions and swap it on the server. To make sure a program never runs with the wrong key, send the `X-API-Mode` header with `sandbox` or `production`: if it does not match the key's mode, the request is refused with `invalid_request` and the reason `mode_mismatch`. Send and check each code with the same key.

Rather try it without writing code?
Every operation in the API documentation has a Playground button: pick the operation, paste the Sandbox key and see the response in your browser. The key stays only in the page's memory.

RECOMMENDED

Want to try SMS verification without using credits? Create your Sandbox key.

See the API page →

SEE ALSO

- **SMS verification API: we generate, send and check the code, and your system keeps only the id**
- **Adding SMS verification to your site: PHP, Node.js and Python examples**
- **Verification code statuses: pending, verified, expired, failed and cancelled**
- **Reading the verification API's answers: valid false, 409, 422 and 429**
- **Sending SMS through the API: key, permissions, Sandbox and your first request**
- **An API key or token has leaked: what to do, in the right order**

interweb

Interweb Mozambique

RLX Holdings, S.A. · Group RLX Holdings

TAX ID (NUIT)	401 191 607
LEGAL ENTITY NO. (NUEL)	105 045 974
ECONOMIC ACTIVITY CODE (CAE)	62022
REGISTERED OFFICE	Bairro 7 de Setembro Vilankulo 1304 · CEP 0411-02, Inhambane, Mozambique
CONTACT	info@interweb.co.mz · +258 84 635 1111 · +258 86 635 1111 · www.interweb.co.mz
OPENING HOURS	Monday to Friday - 08 AM to 04 PM Saturday & Sunday - Closed

Online version of this document

RLX-MZ-KB-2652 · v2026.11.13 · Generated on 13/11/2026 19:30